Radboud University Nijmegen

Faculty of Science

Text-Independent Speaker Recognition

based on DNN embeddings

Thesis MSc Computing Science

Author:
Olga Vishnyakova

Supervisor:
dr. David van Leeuwen

Second reader:
dr. Louis ten Bosch

July 2020

Contents

1 Introduction **2**

2 Background **3**

3 Methods **4**
3.1 Speaker recognition model architecture 4
3.2 Pre-processing 5
3.3 Feature extraction 6
3.4 Robustness to environment 8
3.5 Data augmentation 8
3.6 DNN Architecture 11
3.7 Losses 11
3.8 Backend 12
3.9 Evaluation 13

4 Experimental setup **15**
4.1 Datasets 15
4.2 Evaluation metrics 15
4.3 Data preprocessing and feature extraction 16
4.4 Architecture 16
4.5 Backend 17
4.6 Training 17

5 Results **18**
5.1 System evaluation on MFCCs 18
5.2 System evaluation on log mel filterbanks 19
5.3 Visualizing bottleneck features 21

6 Discussion **24**

7 Conclusion **24**

8 Acknowledgments **24**

1 Introduction

The aim of a speaker recognition algorithm is to determine whether an utterance belongs to a speaker that claims a certain identity. This algorithm could require a person to utter a specific phrase or make a decision based on an arbitrary statement. The first approach is called text-dependent recognition whereas the second one is referred as text-independent. Despite performance achievements, this topic remains a very active research area due to development of neural network topology and the availability of large-scale free datasets.

Regardless of chosen approach, the standard speaker recognition process involves three following main steps [1]. The first stage is *training*, aiming to determine a compact speaker representation from given utterances with known identity. Thus, embeddings are utilized for mapping utterances to speaker feature vectors while distances in constructed feature space correspond to dissimilarity between the voice characteristics of the individuals [2]. The main goal of this step is to train a speaker recognition model able to generalize previously unseen speakers. Further, representations produced by the model are used as inputs for a scoring function. At the second stage, which is known as *enrollment*, a pre-trained model is applied to define the speaker manifold for new individuals. During the last *evaluation* step, a decision is made on the claimed identity by comparing a scoring function to a threshold. Specifically, this function estimates a score between compact representation of a test utterance and enrolled speaker model. Examples for this function are cosine distance scoring or Probabilistic Linear Discriminant Analysis [3].

Most state of the art speaker recognition systems consist of an embedding extraction front-end system followed by a separate backend that is used to compare pairs of embeddings [4]. More recent approaches propose an end-to-end system that jointly learns utterance representations along with a similarity metric [1]. Although the latter allows directly optimize similarity, it requires extra in-domain training speakers which might be challenging for some practical applications [5].

Numerous studies have achieved remarkable gains in automatic speaker recognition mostly due to using deep neural networks (DNN) [6]. However, the task remains challenging especially under noisy and unconstrained conditions [7]. Different neural network (NN) topologies are proposed for speaker recognition models that tend to have deeper and deeper architecture along with more sophisticated losses. Since DNN-based systems are trained on a large speech corpus that is highly time-consuming and requires a significant amount of computational resources that usually involve a GPU, the dependence of performance on NN model capacity becomes interesting to analyze [8].

This work proposes a speaker embedding system for text-independent speaker recognition based on Convolutional neural network (CNN). This thesis examines NN topology, backends and frame-level input features. Part of the aim of this project is to investigate how the number of NN parameters influences on the quality of the recognition. The specific objective of this study is to demonstrate how bottleneck features extracted from the embedding layer contain sufficient information to provide clustering of unseen speakers. Besides performance metrics, this could be considered as an addition evidence of the generalization ability of the model.

2 Background

Over the past years, a dominant approach for speaker recognition was Joint Factor Analysis (JFA) applied to high-dimensional Gaussian Mixture Model (GMM) supervectors originally proposed by Patrick Kenny *et al.*[9]. This method allows to model the inter-speaker variability as well as compensate for channel and session variability by defining two distinct spaces [10, 11]. The main idea behind the technique is mapping variable-length utterances into a fixed-length vectors preserving the speaker information.

Further development of this method led to front-end factor analysis technique, termed 'i-vector' where `'i'` stands for intermediate-size vector. In contrast to the previous method, the proposed approach defines only a single low-dimensional total-variability space that models speaker and channel variabilities concurrently [12]. Besides explicit advantages of the technique, such as reducing complexity of estimating two separate spaces, [10] showed that it is less dependent on score normalization. Moreover, Ahilan Kanagasundaram *et al.*[13] successfully applied this powerful tool to short utterances (less than 10 seconds). For the next several years, there has been an increasing amount of research into speaker recognition based on i-vector approach that showed the state-of-the-art performance. Much of this progress has come from employing session compensation techniques such as Probabilistic Linear Discriminant Analysis (PLDA) [3] after the i-vector extraction [13]. Further experiments revealed benefits utilizing linear discriminant analysis (LDA) [14] before PLDA backend. Obtained low-dimensional representations of feature vectors suppress irrelevant directions caused by noisy conditions or channel distortions [15].

In the last few years, i-vectors were replaced with embeddings obtained from feed-forward DNN with fixed-length inputs. These speaker-discriminative features are known as 'd-vector' [16, 17]. The further improvement resulted in 'x-vector' estimated on variable-length acoustic segments. Although results between i-vecors and x-vectors were comparable on long speech segments (more than 10 seconds), the latter significantly outperformed on short duration test condition [5, 18].

Different architectures for DNN were proposed to obtain speaker representation. For example, a long short-term memory recurrent neural network (LSTM) with generalized end-to-end loss was presented in [19]. In [20] CNN based on VGG architecture [21] was employed for embedding that used a Siamese network trained with the contrastive loss. Most state-of-the-art NN architectures utilized for speaker recognition task are based on 1D CNN and modifications of 2D NN with skip connections. A well-known example of the first type of architecture is the Time-Delay acoustic model (TDNN) [5]. This network includes frame-level layers, statistics pooling that estimates mean and standard deviation over input frame features, segment-level layers and a final softmax where outputs correspond to speaker probabilities. The most widely used 2D architecture is ResNet [22]. In fact, recent publications have revealed that modified ResNets outperformed the TDNN-based systems [23, 24].

There have been several papers that experiment with modifications of the ResNet architecture. The work [2] reported experiments with ResNet and stacked gated recurrent unit (GRU) layers. Wan *et al.* in [25] proposed a recognition deep network that uses Thin ResNet and a dictionary-based trainable aggregation layers: NetVLAD and NetVLAD with ghost clusters (GhostVLAD) [25, 26]. Vector of Locally Aggregated Descriptors (VLAD) is a pooling method that was originally proposed to obtain a compact image representation [27]. A generalized VLAD layer (NetVLAD) was further utilized

as a pooling layer to construct the utterance-level representation [25].

A number of studies have suggested improvements that raised the performance of the speaker recognition model by using margin losses that encourage interclass separability and impose a margin between classes. Recently, [28] examined speaker recognition systems trained with angular, additive margin and additive angular margin losses. In order to encourages intraclass compactness, Joon Son Chung *et al.* [29] utilized a contrastive loss that penalises the distance between negative pairs of utterances (utterances belong to different speakers). Triplet loss was also used in [30] on a internal datasets where Zhang *et al.* employed fixed dimensional spectrogram as the input to CNN.

Commonly used architecture for speaker recognition system consists of DNN-based front-end trained to optimize the classification loss and separate backend to compare a pair of embeddings. Most common is to use LDA projections of x-vectors followed by PLDA scoring [5, 24]. The recently introduced end-to-end architecture combines front-end and backed parts that learns embeddings and a similarity metric simultaneously. Georg Heigold *et al.* in [1] demonstrated the efficiency of this approach over traditional i-vector baseline on the 'Ok Google' benchmark [31].

While most of state-of-the-art systems use hand-crafted front-end features, such as filter bank coefficients or Mel-Frequency Cepstral coefficients, new works have established that first CNNs layers can perform the role of the feature extraction. Thus, such systems could operate on spectral representation or raw waveform data. For instance, Joon Son Chung *et al.* in [29] applied CNN-based system to spectrograms. Raw speech samples were used in [32] as inputs to feed-forward NN. The proponents of these approaches compare feeding raw audio with pixels for images, which has become a standard for image recognition tasks. Experiments with SincNet architecture revealed that engineered features smooth the speech spectrum and hamper detection of narrow-band speaker characteristics such as pitch or formants [33].

3 Methods

3.1 Speaker recognition model architecture

This work introduces a model that describes a standard speaker recognition system. The core architecture of the system is illustrated in Figure 1. During training stage, spectral features are extracted from fixed-time duration segments selected from train utterances and fed to the DNN to obtain speaker representations. In general, new speakers provide several utterances; the final representations are obtained by averaging of vectors corresponding for each speech fragment. Since most recent Speaker Recognition Challenges provide test pairs to estimate a performance of pre-trained models, the number of utterances used in this work for enrollment stage is restricted to a single speech segment. Thus, during enrollment and testings steps, N-dimensional speaker representations are extracted using the pre-trained model and compared using a scoring function. Decision whether each test pair of utterances are from the same or different is made by comparing with the threshold that corresponds to the estimated equal error rate (Section 3.9).

The speaker recognition system proposed in this study uses a four-stage pipeline. First, frame-level acoustic features are computed from given utterances. These short-time content is used as input to a feed-forward DNN. The fixed-length segments are kept short (less than 4 seconds) for training to avoid overfitting. Similar to Snyder *et al.* in [34], the end-to-end approach is split in two sequential stages: front-end DNN and

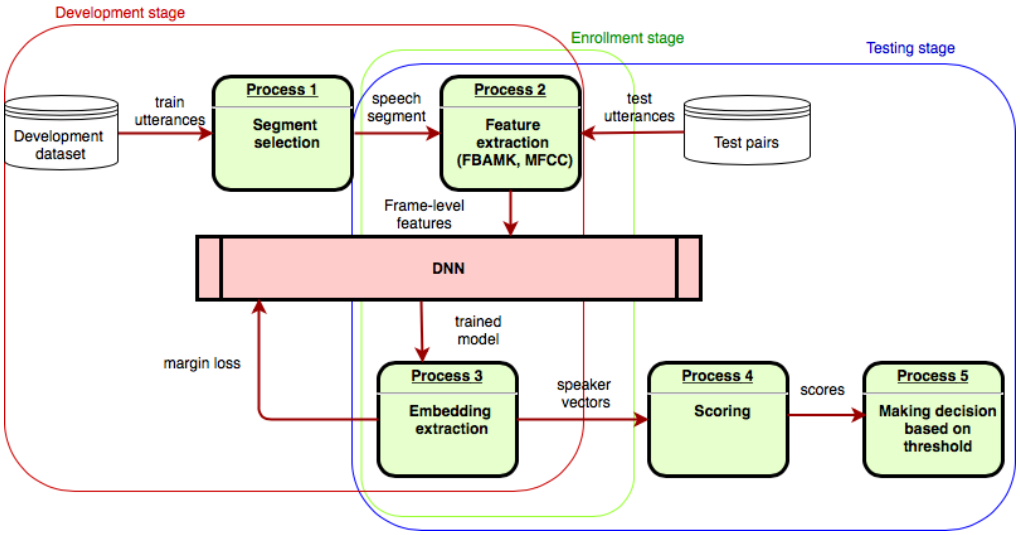


Figure 1: The architecture of DNN-based system for text-independent speaker recognition.

PLDA-based backend. The architecture of the front-end system is shown in Figure 2. The main argument for chosen architecture is that publicly available datasets do not contain sufficient amount of in-domain speakers which is required to train end-to-end systems. Next, the DNN is trained to optimize the classification loss on the training set. After that, embeddings are extracted from a hidden utterance-level layer. Finally, a separately trained PLDA-based backend is employed to evaluate the similarity score.

3.2 Pre-processing

The silence and non-speech segments removal is a crucial pre-processing step that allows to exclude a part from data that does not carry any information about speakers. Hence, as a result, this increases a performance of a recognition system. Speech Activity Detection (SAD) is a technique used to determine speech and non-speech intervals. Since the energy of the non-speech interval is much lower, in a simplest way, a sufficient statistic extracted from a short interval (up to 20 ms) could be compared with a threshold to classify a given frame [35]. Although an accurate detection is a challenging task under noise conditions, a simple energy-based SAD showed a good performance for DNN-based systems [24]. Common approaches to estimate a signal energy are the short-time energy, which represents the average sum of the squares of the samples amplitude, and the root mean square energy (RMSE). Short-time energy statictic is used for proposed model.

Let $u(t)$ be a speech signal and N the length of the frame. Then, short-time energy of the frame i is defined as:

$$E_i = \frac{1}{N} \sum_{t=N(i-1)+1}^{iN} u^2(t) \tag{1}$$

Let E_{max} represents the maximum short-time energy of the utterance. Let θ be a threshold in decibels below zero. Then SAD denotes the frames an active if E_i lies

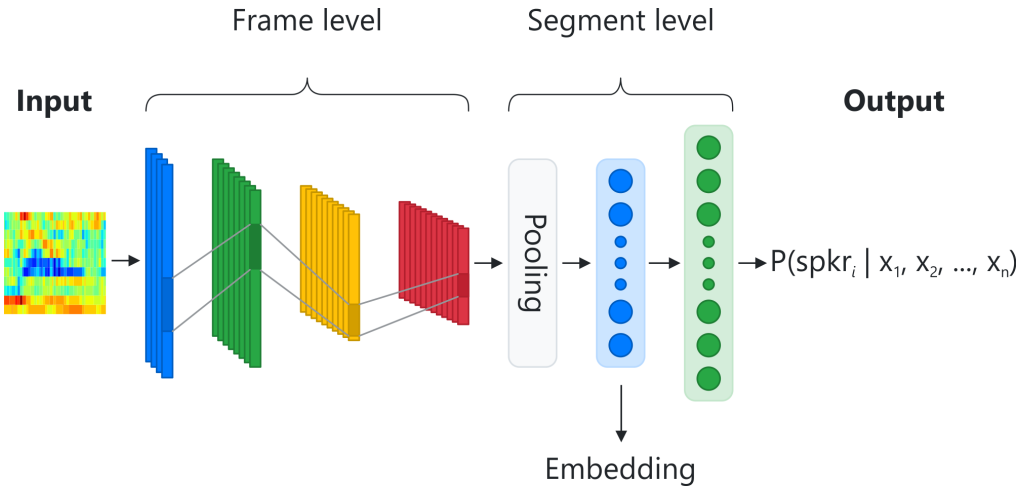


Figure 2: The architecture of the front-end DNN-based system

.

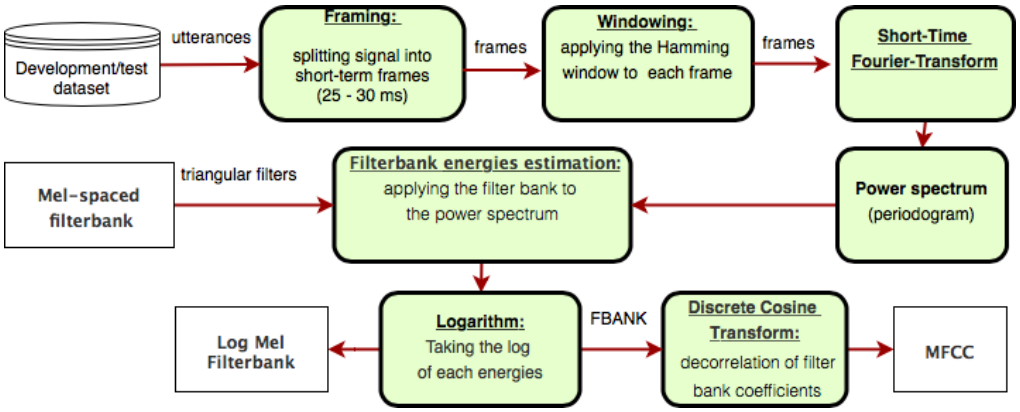


Figure 3: The feature extraction pipeline

.

above the threshold θ estimated in dB:

$$10 \log_{10} \frac{E_i}{E_{max}} > \theta \tag{2}$$

3.3 Feature extraction

Traditionally, Mel-Frequency Cepstral Coefficients (MFCCs) have been employed as engineered features [5, 34]. Although most research on speaker recognition task has been carried out using MFCC, further exploration of the DNN-based models has revealed better performance utilising log-filterbank [1, 24]. The differential and acceleration coefficients, also known as deltas and delta-deltas, usually supplement feature vector. For the first part of experiments, this study uses MFCC and deltas in order to construct a frame representation, whereas for the second part log-filterbanks are used. The feature extraction pipeline is illustrated in Figure 3 and described below.

In order to obtain MFCC, an input signal goes through the number of steps [36]. First, a utterance is split into short-time overlapping frames. The main reason for this

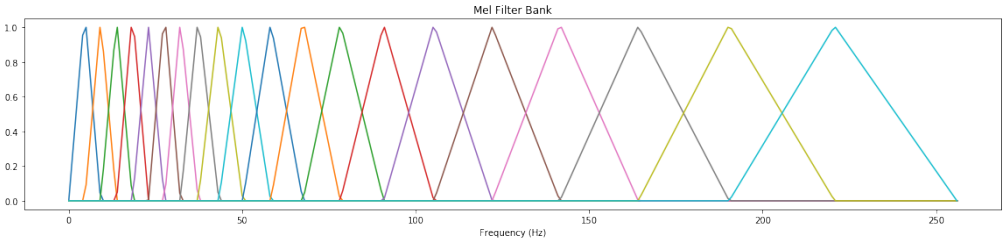


Figure 4: Mel filterbank

.

is a variability of speech signals. Since the Fourier transform is able to process periodic signals, an underlying assumption behind the framing is that signal frequencies are stationary over a short period of time ($25 - 30$ ms and a 10 ms shift). The periodicity condition requires additional step such as windowing applied to each frame to reduce ripple in the spectrum. The common choice is Hamming window that has the following form:

$$w(t) = 0.54 - 0.46 \cdot \cos \frac{2\pi t}{N-1} \tag{3}$$

where, $0 \leq t \leq N-1$, N is the window length.

The next step in the MFCC estimation pipeline is Short-Time Fourier Transform (STFT) followed by the power spectrum. Let x[t], w[t] be an input signal and a window function respectively. Then, STFT and the power spectral could be found as [37]:

$$\begin{aligned} STFT(\tau, \omega) \equiv X(\tau, \omega) = \sum_{-\infty}^{\infty} x[t] w[t-\tau] e^{-i\omega t} \\ S(\tau, \omega) = |X(\tau, \omega)|^2 \end{aligned} \tag{4}$$

There are a number of studies suggested that human perception of frequencies is highly nonlinear [38]. In fact, human ear is more sensitive to variations in low frequency regions. In order to address this issue, the mel scale was proposed that is linearly spaced in low frequencies and logarithmic in high frequency regions. Mel-filterbank is illustrated in Figure 4. Linear frequencies (Hz) could be converted to mel scale using a following rule [39]:

$$f_{mel} = 2595 \cdot \log_{10}(1 + \frac{f_{lin}}{700}) \tag{5}$$

The next step of the pipeline is to apply mel filterbanks to the power spectrum to estimate the total energy within the certain filter band. Due to the fact that the human ear response in nonlinear way not only to frequency but also to amplitude, the logarithm is taken after filtering that results to log-filter banks, often denoted as FBank. To decorrelate filterbank coefficients, the Discrete Cosine Transform (DCT) is applied which resulted in MFCCs by taking the first few coefficients [40]. The outputs of each step are shown in Figure 5.

DCT type-II is used in this pipeline that could be defined in a following way. Let M be the number of filters in the mel filter bank, K be the number of cepstral coefficients and X_i be the output of i-th filter. Then MFCCs could be computed as:

$$C_k = \alpha_k \sum_{i=1}^{M} X_i \cos \frac{\pi k(2i+1)}{2M}, k = 0, 1, 2...K \tag{6}$$

where $\alpha_k = \frac{\sqrt{2}}{M}$ for $k = 0$, $\alpha_k = \frac{2}{M}$ for $k > 0$.

Once MFCCs are estimated, the delta and delta-delta coefficients are computed to preserve the dynamic information of the signal. Delta-Delta coefficients are calculated in the same way using deltas as input. The differential coefficients are obtained as:

$$d_t = \frac{\sum_{n=1}^{N} n(C_{t+n} - C_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{7}$$

where N is typically set to 2.

3.4 Robustness to environment

Previous research has established that data normalization is crucial step before training a speaker recognition model that allows to rise classification accuracy [20]. Thus, inputs are normalized on per utterance basis before feeding to NN. The results of normalization applied for one of the test utterances are shown in Figure 6.

In addition to data preprocessing, batch normalization is used inside NN. It forces activations to have zero mean and unit standard deviation across the mini-batch. As a consequence, that leads to faster convergence to global minima at error surface [41]. This is especially a case for large mini-batch sizes. Thus, this study employs a frame-level features normalization on a per-utterance basis and batch normalization inside the network.

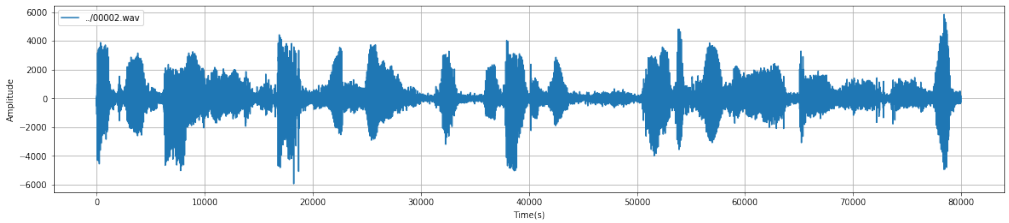
3.5 Data augmentation

In order to avoid overfitting, increase generalization ability of the model and rise robustness against channel variation, data augmentation is widely used for NN training. This implies creating additional transformed training data. Traditionally, modifications are applied in the time domain by adding noise, background music or reverberation [34]. The main drawback of this approach is storing augmented copies of original utterances. This results in increasing training dataset by 50% or 100% depending on augmentation strategy.

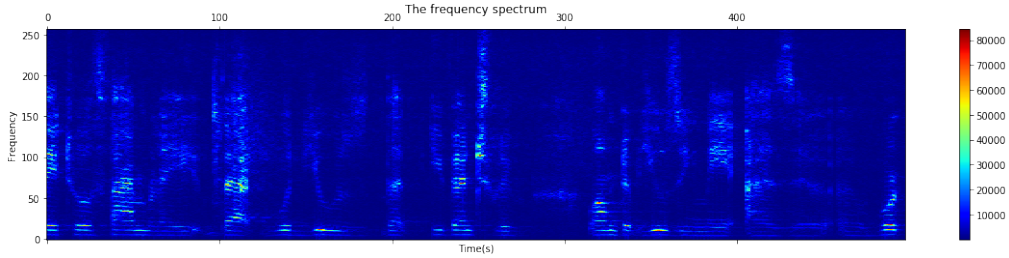
Recently, [42] proposed a on-the-fly augmentation method that applied directly to the a log mel spectrogram, rather than to a raw audio input. This technique was selected for its simplicity and effectiveness reported in [24]. Another advantage that it does not require any noise datasets. In addition, since spectrum augmentation modifies inputs online during training, there is no need to store addition transformed data.

This study applies two types of spectrum augmentation. First kind of deformations is a time masking. Let τ be the number of time steps and T be a pre-defined time mask parameter. Then, consecutive time steps $[t_0, t_0 + t)$ are set to spectrogram mean where $t \in \mathcal{U}(0, T)$, $t_0 \in [0, \tau - f)$. In case of a normalized input, this is the same as to set values to zero.

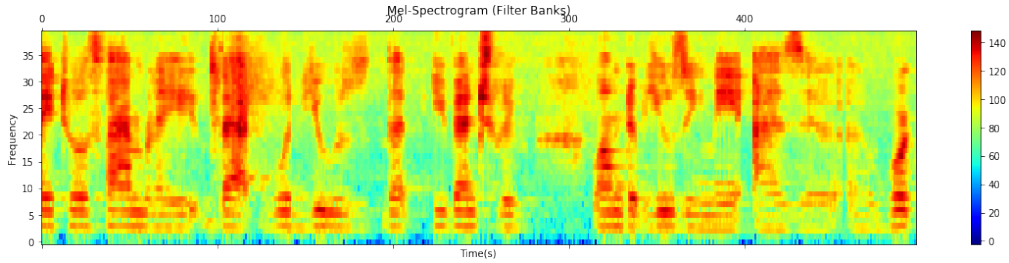
The second type of spectrum augmentation used for experiments is a frequency masking. Similarly, let ν be the number of mel frequency channels and F be a pre-defined frequency mask parameter. Then, consecutive mel frequency channels $[f_0, f_0 + f)$ are set to spectrogram mean where $f \in \mathcal{U}(0, F)$, $f_0 \in [0, \nu - f)$. Figure 7 shows examples of the spectrum augmentations applied to a single log mel spectrogram.



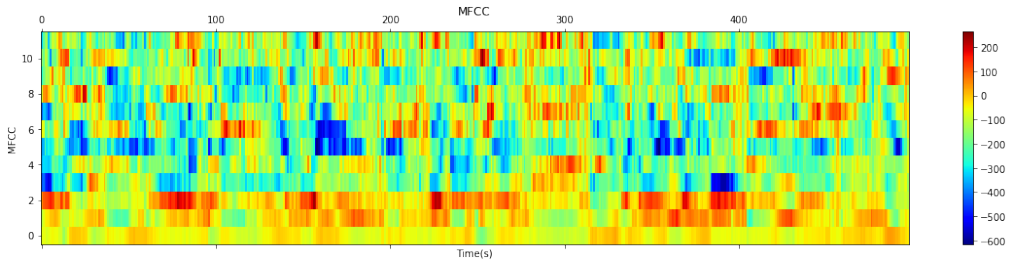
(a) Utterance in the time domain



(b) Spectrogram of the signal

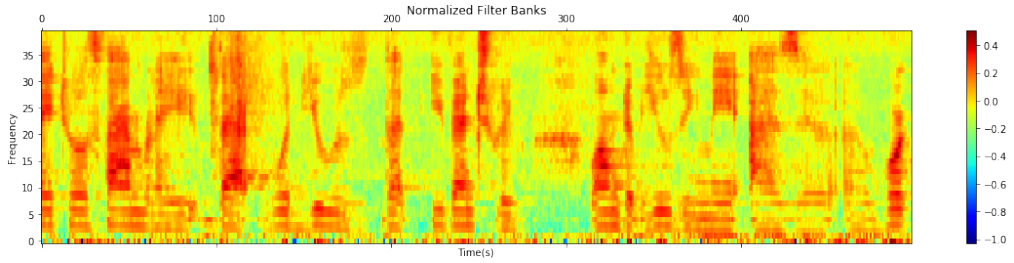


(c) Log-filterbank of the signal

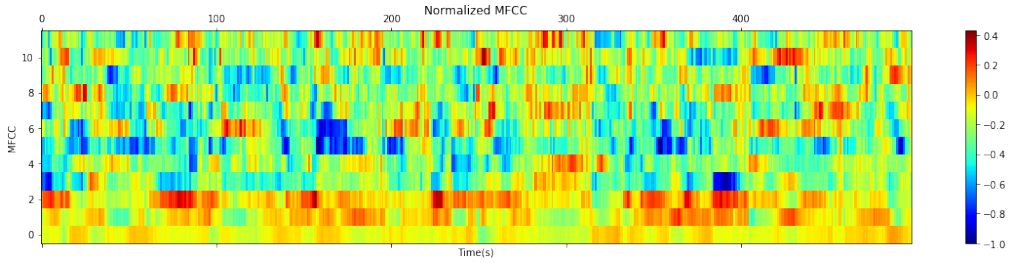


(d) MFCC

Figure 5: The feature extraction process. Example of processing of one of the test utterances.

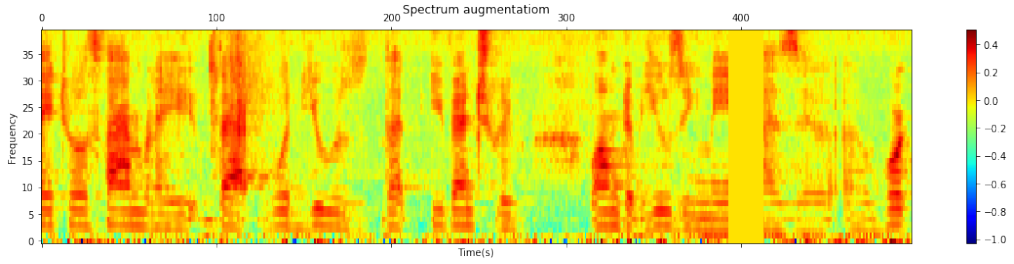


(a) Normilized MFCC

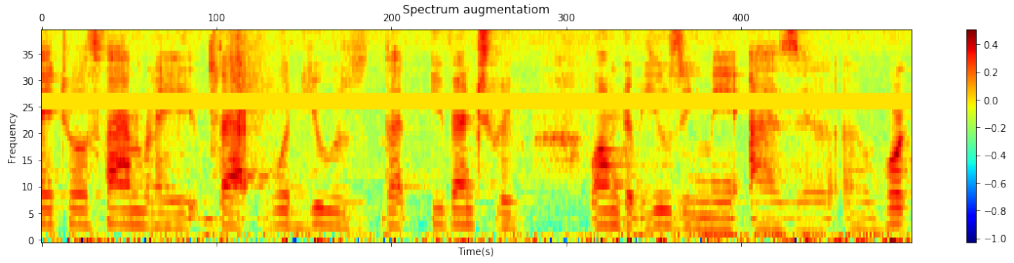


(b) Normilized MFCC

Figure 6: Normalized input features.



(a) Time masking



(b) Frequency masking

Figure 7: Augmentations applied to a single log mel spectrogram.

3.6 DNN Architecture

The architecture of all speaker recognition models that was investigated in this study is based on CNN. Despite of the variability of layers and capacity, all systems contain same important elements: a part that operate on a frame level, pooling, a segment-level part which includes a fully-connected layer to compute embeddings. The main idea behind the pooling layer is to aggregate the information over all input frames by averaging (Average Pooling) or computing mean a standard deviation (Statistics pooling). The former approach is commonly used for CNN while the latter for TDNN. Moreover, pooling encourages the utterance representation become roughly invariant to small translations[43].

A DNN could be used for computing features for a second classifier, for instance, PLDA or another DNN [44]. This could be achieved by using activations of one of the hidden layer as new feature vectors. Although any layer could act as a bottleneck layer, it has been demonstrated that the utterance-level representation outperforms in comparison with frame-level embedding[1]. For every architecture, a layer after pooling is utilized in this work as a bottleneck layer to extract speaker representations for all models. These embeddings become inputs to a backend.

In this study, d-vector approach based on VGG architecture proposed by [21] is used as the baseline. The convolutional layers of this model mostly have 3×3 filters. This plain CCN demonstrated good generalization ability and achieved state-of-the-art results in the classification task in the ImageNet Challenge 2014.

This thesis mainly focuses on ResNet-based architecture. Thus, all speaker recognition systems use a residual learning framework that is claimed to be easier to optimize than plain models. The main reason for that is imposing shortcut connections that help to handle with the degradation of training accuracy issue. [22]. ResNet consists of two types of blocks that consist of several convolutional layers supplemented by skip connections. The choice of using a certain type depends on the a ResNet depth. In fact, *'building'* blocks with two 3×3 convolutional layers are used for ResNet18 and ResNet34 topologies, while ResNet50 and deeper contain co-called *'bottleneck'* blocks that consists of tree layers. The structure of blocks is shown in Figure 8. The 1×1 layers bottleneck blocks are responsible for reducing dimensions before the expensive 3×3 convolutions and then restoring dimensions.

3.7 Losses

The most widely used loss function utilized for a multi-class classification task is Softmax. Let C be a number of classes and d be a presoftmax layer dimension. Then, Softmax is defined as follows:

$$L_{\text{softmax}} = -\log \frac{e^{W_y^T x + b_y}}{\sum_{j=1}^{c} e^{W_j^T x + b_j}} \tag{8}$$

where $x \in \mathbb{R}^d$, denotes the input of the sample that belongs to the y-th class. $W \in \mathbb{R}^{d \times c}$ and $b_j \in \mathbb{R}^C$ are the weight matrix and the bias term in the presoftmax layer respectively [45].

Despite of fast convergence and good performance for the closed-set classification, there are some drawbacks. This loss function does not explicitly promote inter-class separability. A number of studies have established that activations sampled from the projection layer are not discriminative enough for the open-set recognition problem [28, 46, 47].

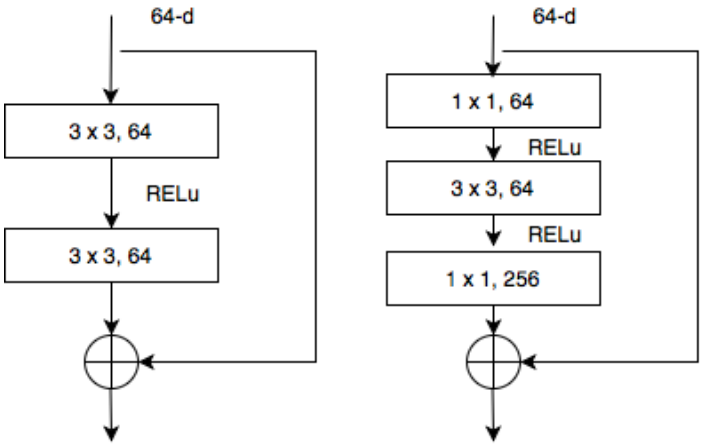


Figure 8: ResNet block. Left: a building block. Right: a bottleneck building block.

To handle the problem, margin-softmax losses are proposed that incorporate margins between classes, such as the angular softmax (SphereFace) [47], CosFace [46] and Additive Angular Margin Loss (AAM-loss or ArcFace) [45]. All magrin-softmax losses learn angularly discriminative features that minimize intra-class variance and maximize inter-class variance. AAM-loss was selected for this study based on reported performance.

First, let the bias $b_j = 0$ be fixed in Eq.(8). After that, the output of the last layer $W_j^T x$ could be written as:

$$W_j^T x = \left\|W_j\right\| \|x\| \cos\theta_j \tag{9}$$

where θ_j denotes the angle between the individual weight and the feature. If weights W_j and embedding features x are L_2-normalized, the angle could be calculated as:

$$\theta = \arccos W_j^T x$$

After re-scaling features to make them distributed on a hypersphere with a radius s, the AAM-loss can be defined as:

$$L_{\mathrm{aam-loss}} = -\log \frac{e^{s(\cos(\theta_y + m))}}{e^{s(\cos(\theta_y + m))} + \sum_{j=1, j\neq y}^{c} e^{s(\cos(\theta_j))}} \tag{10}$$

where m is an angular margin penalty.

3.8 Backend

Traditionally, cosine similarity scoring has been utilized for CNN-based model to compare pairs of embeddings. It was shown that imposing a space between classes by training with margin-losses led to good generalization ability. Thus, cosine distance could be a simple choice [24] for scoring. However, resent work have report better performance with a separately trained backend model based on PLDA. Especially it is important when a severe domain-shift between train and test takes place [23].

This study investigates both of the proposed backends. Once the NN model is trained, activations of the projection layer are computed for each test pair to obtain

embeddings. For the first backend, a finale score is estimated using a simple cosine similarity function as:

$$score(emb_{spk_1}, emb_{spk_2}) = \frac{emb_{spk_1}^T \cdot emb_{spk_2}}{||emb_{spk_1}|| \cdot ||emb_{spk_2}||} \tag{11}$$

The PLDA technique was firstly introduced for a face recognition [3]. Further, it demonstrated effectiveness for speaker recognition tasks when it was applied to the i-vectors. The simplest PLDA model is a linear model with Gaussian noise. Thus, x-vector generation process could be modeled as a sum of factors:

$$x_{ij} = \mu + Fh_i + Gw_{ij} + \epsilon_{ij} \tag{12}$$

where x_{ij} denotes j-th embedding of i-th speaker; μ is a mean of the training dataset; h_i is a latent speaker factor; w_{ij} is an inter-session component, and ϵ_{ij} is the residual noise. This model contains a speaker-dependents part $\mu + Fh_i$, which depends only on identity of the individual and determines between-speaker variation [48]. The columns of F are basis vectors for the between-individual subspace. The second part $Gw_{ij} + \epsilon_{ij}$ of Eq.12 is a noise component, which depends on a single utterance and determines within-speaker variability. The columns of G are basis vectors for the within-individual subspace.

Simple Gaussian PLDA assumes priors for model parameters to be normally distributes. Thus:

$$\begin{aligned} Pr(x_{ij}|h_i, w_{ij}, \theta) &= \mathcal{G}_x[\mu + Fh_i + Gw_{ij}, \Sigma] \\ Pr(h_i) &= \mathcal{G}_h[0, I] \\ Pr(w_{ij}) &= \mathcal{G}_w[0, I] \end{aligned} \tag{13}$$

where Σ is a covariance of ϵ_{ij}; I is the identity matrix, and $\mathcal{G}$ denotes the Gaussian distribution.

During training, the PLDA model learns the parameters $\theta = \{\mu, F, G, \Sigma\}$ from the labelled embeddings x_{ij} that maximize the likelihood of observed data assuming that same value for the latent variable h_i corresponds to the same speakers. In the testing stage, the speaker recognition log-likelihood ratio score is calculated between two hypothesis: two utterances were generated from the same speaker factor h_i, utterances were generated from the different speaker factor h_i.

To make learning process more efficient, embeddings extracted from the trained NN undergo several transformations before feeding to the PLDA model. The dimensionality of mean-normilized embeddings is reduces using LDA. To preserve an assumption regarding Gaussian priors for the PLDA model parameters, [49] proposed applying length normalization. In order to convert PLDA scores into probabilities, a calibration is required. This study uses Sigmoid transformation defined by following:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{14}$$

This transformation converts log-likelihood ratios to the unit interval as it required for the VoxCeleb Speaker Recognition Challenge.

3.9 Evaluation

Two types of errors are associated with the speaker recognition task. The former is false acceptance when the access is granted for an imposter, and the latter is false rejection

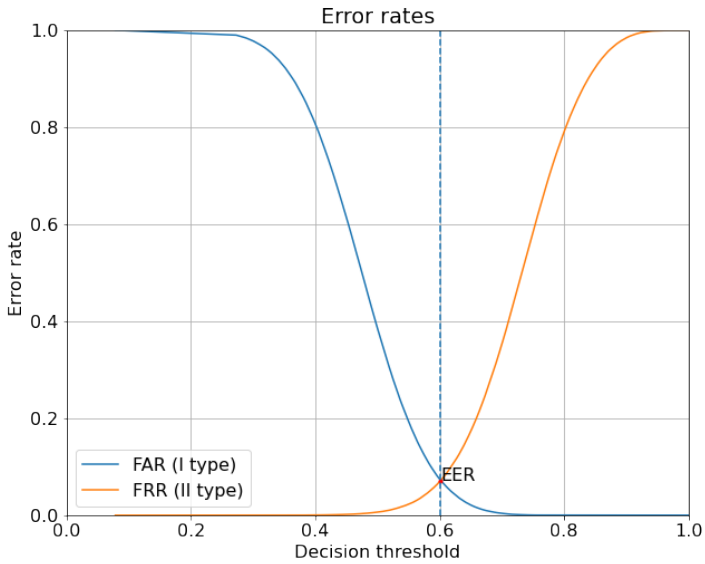


Figure 9: A trade-off between False Acceptance Rate and False Rejection Rate. A cross point is EER.

when access to a genuine speaker is denied. The frequency with which the first type of errors occurs is called False Acceptance Rate (FAR) or False Alarm probability. It is defined as the number of imposters accepted divided by the total number of imposter attempts. The frequency of the former is called False Rejection Rate (FRR) or Miss probability which is defined as a ratio between the number of rejected genuine speakers and the total number of legitimate attempts [50].

The FAR and FRR rates change opposite way: while the discrimination threshold rises, the FAR is decreased, but the FRR is increased. This trade-off is illustrated in Figure 9. The threshold value for equal FAR and FRR is called the Equal Error Rate (EER) (it is shown as a cross point between two rates in Figure 9). EER is a common metric that is used to report speaker recognition system performance.

For real application the cost associated errors could be explicitly determined. In order to take into consideration penalties that are specific for a particular application, the detection cost is utilised. This cost function constitutes a weighted sum between false rejections and false acceptance rates and is defined as follows:

$$C_{det} = C_{FR} \cdot P_{FR} \cdot P_{target} + C_{FA} \cdot P_{FA} \cdot (1 - P_{target}) \tag{15}$$

where C_{FR}, C_{FA} are costs of false rejection and false acceptance, P_{target} is a prior target probability. P_{FR} and P_{FA} are measured probabilities that depends on the acceptance threshold. The minimum of C_{det} values computed for the range of threshold levels is the second widely used metric for speaker recognition tasks. Both of these evaluation metrics, EER and C_{det}^{min}, are employed to report system performance.

Traditionally, The Receiver Operating Characteristic (ROC) curve is utilized to illustrate the prediction performance of a binary classifier that plots the probability of correct acceptance (1-FRR) against FPR at various decision threshold values. To compare performance of speaker recognition model, it is more common to use Detection

stat	**total**	VC1 Dev	VC1 Test	VC2 Dev	VC2 Test
# of speakers	7 363	1 211	40	5 994	118
# of videos	172 976	21 819	677	145 569	4 911
# of utterances	1 281 762	148 642	4 874	1 092 009	36 237
avg # of utterances	-	116	-	185	-
avg length of utterances	-	8.2	-	7.8	-

Table 1: VoxCeleb datasets details. VC1 and VC2 denote VoxCeleb1 and VoxCeleb2 respectively.

Error Trade-off (DET) Curve which is similar to ROC curve, but is more linear due to axis warping. Specifically, axis are non-linearly scaled to highlight the differences between models in the critical operating region [51].

Let $x = P_{fa}$ and $y = 1 - P_{miss}$ be the the horizontal axis and the vertical axis for the ROC curve, where P_{fa} is the false alarm probability and P_{miss} is the the miss probability. Then, axis for DET curve could be obtained by normal deviate scaling: $x = \text{probit}(P_{fa})$ and $y = \text{probit}(P_{miss})$ where the probit function maps the unit interval $[0, 1]$ to $[-\infty, \infty]$ and is defined as:

$$\text{probit}(p) = \sqrt{2}\ \text{erf}^{-1}(2p - 1) \tag{16}$$

where erf^{-1} is the inverse error function.

4 Experimental setup

4.1 Datasets

The experiments are carried out on the VoxCeleb2 and VoxCeleb1 datasets [20]. Both datasets are contain utterances extracted from YouTube videos. The audio channels of all video have been converted to single channel with a sample rate of 16kHz. Datasets are gender balanced and do not overlap with the identities. The training part of VoxCeleb2 dataset has 5994 speakers and more than 1.2M speech segments. The average length of utterances is 8 seconds. Detailed information about datasets is presented in Table 1.

The VoxCeleb2 train dataset was used for training front-end and back-end models while results are reported on the VoxCeleb1 development or test set. Evaluation lists were provided by the organizers of the VoxCeleb Speaker Recognition Challenge [29]. The VoxCeleb1-O list contains random pairs selected only from the test set with 40 different speakers. In the VoxCeleb1-E, 581 480 pairs were sampled from the entire set with the 1251 identities. The last list, VoxCeleb1-H, consists of 552 536 pairs and compares only speakers with the same gender and nationality which makes evaluation much harder.

4.2 Evaluation metrics

In order to observe the performance of the proposed model, two metrics are utilized for the experiments. These metrics were used in published papers to report results obtained on the dataset and the same evaluation lists. The primary metric is EER. In addition, results are reported using the minimum of the normalized detection cost

Layer	Structure	Output
conv1	3×5, 96, Stride (1,2)	$96 \times 60 \times 149$
mpool1	3×3, Stride 2	$96 \times 29 \times 74$
conv2	3×5, 256, Stride (1,2)	$256 \times 29 \times 74$
conv3	3×3, 256	$256 \times 29 \times 36$
conv4	3×3, 256	$256 \times 29 \times 36$
conv5	3×3, 256	$256 \times 29 \times 36$
mpool5	3×3, Stride 2	$256 \times 14 \times 11$
fc6	256×2048	$2048 \times 1 \times 11$
apool6	-	$2048 \times 1 \times 1$
fc7	2048×512	512
fc8	$512 \times N$	N

Table 2: VGG-M architecture with N classes. 512-dimensional fc7 vectors is used to obtain embeddings. Total number of parameters: 10.5K.

function (minDCF) with $P_{target} = 0.01$ and equal unit weights between misses and false alarms [29].

4.3 Data preprocessing and feature extraction

Two types of engineered features were used for experiments: MFCCs and log filterbanks. Prior to data processing, audio segments with duration less than 4 seconds were excluded from the train dataset. For each utterance from the train and test dataset was applied the same preprocessing pipeline. First, the same energy-based SAD, provided by librosa library [52], filtered out non-speech intervals. The threshold in decibels was set to 30dB.

Both types of the features were calculated with a frame shift of 10ms and frame-length of 25ms. For the MFCC-based experiments, the features are 20-dimensional MFCCs. 20 delta and 20 delta-delta coefficients are appended to construct 60-dimensional feature vector. For the second part of experiments, 80 log filtersbanks were calculated for each frame. After that, obtained features were normalized on per utterance basis to have zero mean and unit variance.

4.4 Architecture

All proposed front-end systems are founded on DNN. For the baseline model, d-vector approach based on VGG-M architecture proposed by [21] with the softmax loss function is used as a front-end model, whereas backend utilizes the cosine similarity between speaker representations. Adjusted to the input size the CNN architecture is specified in Table 2. Good classification performance on large-scale image datasets and the ability to generalize very deep features make a choice of the baseline architecture highly reasonable.

One of the research questions is to investigate how NN capacity influences on the model performance. Hence, three ResNet-based NN are used to compare quality of recognition. A residual learning framework proposed in [22] outperform plain CNN due coping with the degradation problem for very deep models. Since training of extremely deep model is highly time-consuming, performance metrics are estimated on ResNet18 and ResNet-34-based front-end models. To examine how number of filters influences on model performance, two types of ResNet-34 models were trained. The first model has the same filter size as ResNet18; the second model, ResNet-II, has two times more

layer	ResNet-18		ResNet-34-I		ResNet-34-II	
	Structure	Output	Structure	Output	Structure	Output
input	-	$1 \times 60 \times T$	-	$1 \times 60 \times T$	-	$1 \times 60 \times T$
conv1	7×7, 16, Stride1	$16 \times 60 \times T$	7×7, 16, Stride1	$16 \times 60 \times T$	7×7, 16, Stride1	$16 \times 60 \times T$
Block1	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 2$	$16 \times 60 \times T$	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 3$	$16 \times 60 \times T$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$32 \times 60 \times T$
Block2	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 2$	$32 \times 60 \times T/2$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 4$	$32 \times 60 \times T/2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4$	$64 \times 60 \times T/2$
Block3	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$64 \times 60 \times T/4$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 6$	$64 \times 60 \times T/4$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 6$	$128 \times 60 \times T/4$
Block4	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$128 \times 30 \times T/8$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$	$128 \times 30 \times T/8$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$	$256 \times 30 \times T/8$
fc6	128×256	$256 \times 1 \times T/8$	128×256	$256 \times 1 \times T/8$	256×256	$256 \times 1 \times T/8$
apool6	-	$256 \times 1 \times 1$	-	$256 \times 1 \times 1$	-	$256 \times 1 \times 1$
fc7	256×512	512	256×512	512	256×512	512
fc8	$512 \times N$	N	$512 \times N$	N	$512 \times N$	N
# params	1.8K		2.5K		7.4K	

Table 3: ResNet-based architecture. N is the number of classes. 512-dimensional fc7 vectors is used to obtain embeddings. T denotes an input length in frames. Last line is a total number of parameters to train for each network.

filters. The ResNet-based configurations are outlined in Tables 3.

4.5 Backend

As a backend for all models, two the most common approaches are chosen. The first method uses cosine similarity between test embeddings without any preprocessing. In addition, same metrics are computed on centered embeddings. Similar to [53], the training mean was calculated on 500K randomly chosen utterances ($\approx$ a half of the training data).

The second approach employs PLDA. Same 500K utterances were used to train the backend model. First of all, extracted embeddings are centered using same training data mean. After that, feature vectors are projected to M-dimensional representation applying LDA. To define the optimal dimensionality, space size is set to 100%, 75% and 25% of original size (512, 374 and 128 respectively). After length normalization, projections are compared using PLDA scoring. Finally, the Softmax function transforms scores to a probability range [0,1].

For the first part of experiments, the PLDA model is trained on embeddings extracted from short utterances ($\approx$ 3 sec). Same segment size was used for training NN. A final embedding for a single test utterance was obtained by averaging six vectors corresponding to random cuts of an original utterance. For experiments on log mel filterbanks features, embeddings were estimated on full-length utterances.

4.6 Training

Models are trained on short segments that are randomly sampled from original utterances. To control training process and avoid overfitting, 3% of the train set is used as a validation set. Firstly, all networks are trained using Softmax loss function that

shows more stable convergence than margin-softmax losses [53]. After that, the last pre-softmax layer is removed, and networks are fine-tuned with AAM-loss with $s = 30$ and $m = 0.2$ without freezing layers as it was suggested in [28].

All the work was carried out using PyTorch. Each model is trained on GeForce GTX 1080 GPU for 7 epochs for each loss with a batch-size of 64 for VGG, ResNet18, ResNet34-I and 32 for ResNet34-II. Stochastic gradient descent with the moment (0.9) is employed to optimize the network. Learning rate was initially set to 10^{-2} and decreased to 10^{-4} after three epochs. Batch normalisation is applied before ReLU activation function.

5 Results

5.1 System evaluation on MFCCs

In these experiments, performance metrics are estimated on four CNN models and two losses (Softmax and AAM-softmax) on the VoxCeleb-O evaluation list. Besides, four backends are applied to compare the performance: cosine similarity, cosine similarity after centering, cosine similarity score of LDA projections, length-normalized projections modelled by PLDA. Scores are estimated for short segments extracted from each pair and averaged after that. The experiments show that using mean of PLDA scores gives better results that estimate score of means of embeddings. Thus, this approach was used for further experiments. It is worth mentioning that, after training for 7 epochs, accuracy on a validation set is still increasing, but convergence is much slower. Moreover, AMM-loss converges even slowly, probably, due to additional restriction on a margin that should be preserve during training.

The main results obtained on original VoxCeleb1 evaluation list are presented in Table 4. To highlight dependence on volume of the corpus data used for training a network, EERs are estimated on VGG-M-based model trained on VoxCeleb1 and VoxCeleb2 train sets. Since experiments showed that 128-dimensional representation LDA revealed slightly better results, it was used for all further tests. The VoxCeleb-O evaluation list has equal amount of match and non-match pairs. Figure 10 (a) represents histogram of the cosine similarity score distributions for target and non-target pairs evaluated for the ResNet34-II model.

By comparing the different systems trained on VoxCeleb2 train set, the deepest ResNet-based model trained with Softmax achieved the lowest error rate 6.97% even though the number of parameters 29% less than for a baseline model. What stands out in the table is that doubling the number of filters in the building blocks results in 16% better EER and almost three times more parameters to train in comparison to ResNet-I model. However, it more effective than increasing the depth and preserving filter bank size. In fact that gives only 5.4% reduction of EER. From this data, it could be concluded that the PLDA backend shows better results for all models and metrics. In terms of minDCF, ResNet34-II fine-tuned with AAM-loss achieved the best result. However, it is only 1% better than for same model trained with softmax. During the additional experiment, it was observed that extracting embeddings before the RELu activation function is more effective. This is better by 4.4% in EER.

The results of the experiments show that ResNet34-II model outperformed ResNet34-I and ResNet34-II trained with the magrin loss at all operating points. These outcomes are illustrated by the DET curves in Figure 10 (b). Overall, these results indicate that

Model	Loss	Backend	Training set	EER(%)	C_{det}^{min}
VGG-M	softmax	cosine similarity	VoxCeleb1	20.43	-
VGG-M	softmax	cosine similarity	VoxCeleb2	14.69	-
VGG-M	softmax	centering + cos	VoxCeleb2	12.61	-
VGG-M	softmax	LDA + cos	VoxCeleb2	10.58	-
VGG-M	softmax	LDA(128)+PLDA	VoxCeleb2	10.73	-
VGG-M	softmax	LDA(374)+PLDA	VoxCeleb2	10.76	-
VGG-M	softmax	LDA(512)+PLDA	VoxCeleb2	10.79	-
ResNet18	softmax	cosine similarity	VoxCeleb2	11.94	-
ResNet18	softmax	centering + cos	VoxCeleb2	10.36	-
ResNet18	softmax	LDA(128) + cos	VoxCeleb2	10.94	-
ResNet18	softmax	LDA(128)+PLDA	VoxCeleb2	9.2	-
ResNet34-I	softmax	cosine similarity	VoxCeleb2	9.70	-
ResNet34-I	softmax	centering + cos	VoxCeleb2	9.02	0.716
ResNet34-I	softmax	LDA(128)+PLDA	VoxCeleb2	8.71	0.692
ResNet34-II	softmax	cosine similarity	VoxCeleb2	8.54	0.676
ResNet34-II	softmax	centering + cos	VoxCeleb2	7.62	0.665
ResNet34-II	softmax	LDA(128)+PLDA	VoxCeleb2	**7.29**	0.654
ResNet34-II	AAM-loss	cosine similarity	VoxCeleb2	7.71	0.662
ResNet34-II	AAM-loss	centering + cos	VoxCeleb2	7.78	0.658
ResNet34-II	AAM-loss	LDA(128)+PLDA	VoxCeleb2	7.50	**0.647**

Table 4: Experimental results on MFCCs on VoxCeleb-O evaluation list (includes 37720 pairs and 40 speakers).

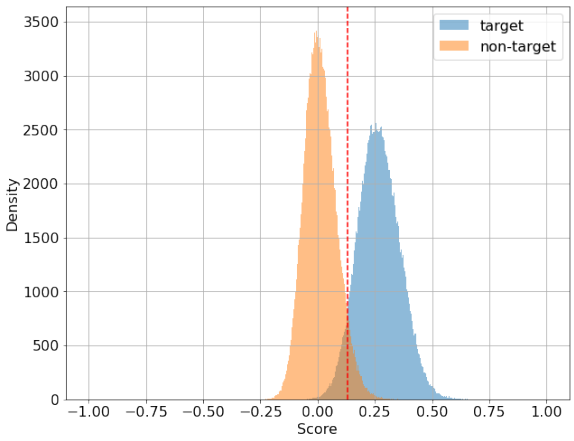
not only capacity, but also architecture play important role.

5.2 System evaluation on log mel filterbanks

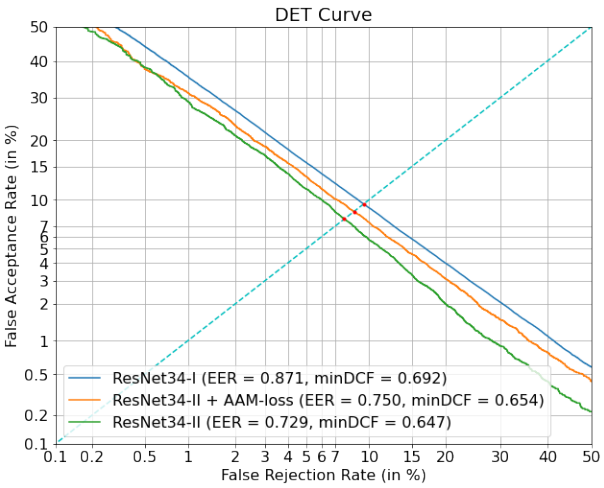
In this section, experiments are carried on mel log filterbanks. The main results obtained on VoxCeleb1 evaluation lists are presented in Table 5. Experimental results presented in Section 5.1 were took into account to achieve better performance. In fact, the ResNet34-II architecture was chosen for all further experiments. In addition, all NN were trained on longer segments ($\approx$ 4 sec); embeddings were extracted on full-length utterances before RELu activation. PLDA was used as a backend model applied to length-normalized 128 projections. The augmentation strategy used for experiments is described in Section 3.5.

For the first experiment, the model was trained without spectrum augmentation. Similar to results obtained on MFCC features, the system with PLDA backend outperform at all operating points. Thus, the PLDA backend was employed for all further experiments. Figure 11(a) illustrates the DET curves for the systems with three different backend: cosine similarity scoring, centered cosine similarity scoring, PLDA backend. Although centered cosine scores are better at the EER point, they are slightly worse than cosine scores at a very low false rejection rate.

As can be seen from the Table 5, spectrum augmentation results in a clear improvement for all evaluation lists. This model achieves an EER of 4.67% on the VoxCeleb-O evaluation list which is 7% better than for a system without augmentation. Same trend is observed for other evaluation lists. In terms of minDCF, system with augmentation is just about 1.5% better. In contract to expectation, a model fine-tuned with the margin

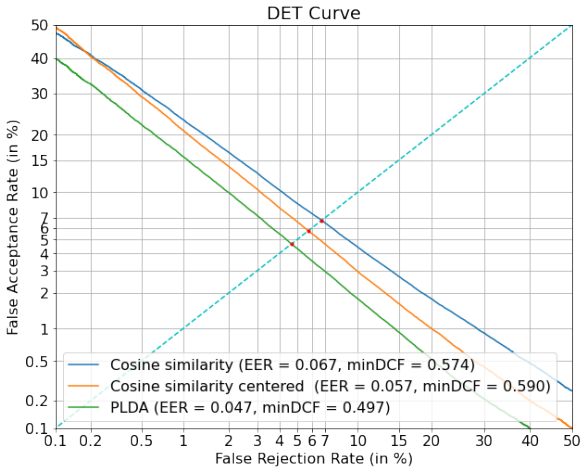


(a) A histogram of the centered cosine similarity score distributions target/non-target for ResNet34-II model trained with softmax.

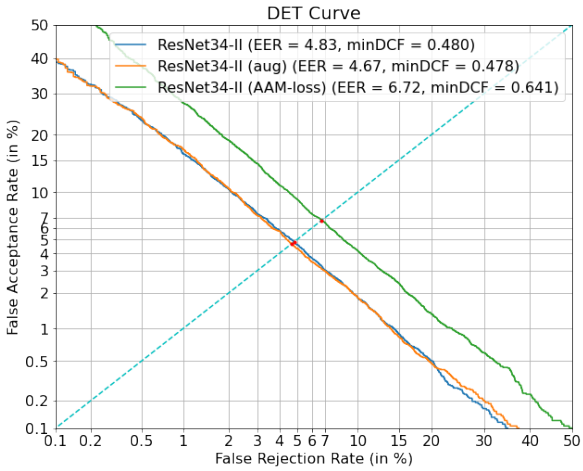


(b) DET curves for ResNet34-based models. EERs are indicated as red dots.

Figure 10: Results for ResNet34-based models.



(a) DET curve for ResNet34-II model trained with Softmax without augmentation evaluated for the VoxCeleb-E list.



(b) DET curves for ResNet34-based models evaluated for the VoxCeleb-O list.

Figure 11: Results for ResNet34-based models.

Model	Loss	Backend	Eval. list	EER	C_{det}^{min}
ResNet34-II	Softmax	LDA+PLDA	VoxCeleb-O	4.83	0.480
ResNet34-II (aug)	Softmax	LDA+PLDA	VoxCeleb-O	**4.67**	**0.478**
ResNet34-II (aug)	AAM-softmax	LDA+PLDA	VoxCeleb-O	6.72	0.641
ResNet34-II	Softmax	LDA+PLDA	VoxCeleb-E	4.67	0.497
ResNet34-II (aug)	Softmax	LDA+PLDA	VoxCeleb-E	**4.58**	**0.493**
ResNet34-II (aug)	AAM-softmax	LDA+PLDA	VoxCeleb-E	7.17	0.688
ResNet34-II	Softmax	LDA+PLDA	VoxCeleb-H	8.16	0.646
ResNet34-II (aug)	Softmax	LDA+PLDA	VoxCeleb-H	**7.91**	**0.644**
ResNet34-II (aug)	AAM-softmax	LDA+PLDA	VoxCeleb-H	12.2	0.796

Table 5: Experimental results on log mel filtebanks features on three evaluation lists: VoxCeleb-O (37 720 pairs and 40 speakers), VoxCeleb-H (set within pairs with the same gender and nationality that includes 552 536 pairs and 1251 speakers), VoxCeleb-E (includes 581 480 pairs and 1251 speakers).

showed much worse performance. Further training for five additional epochs resulted in 33% higher EER. Additional experiment with fixing layers before the pooling layer during fine-tuning, has not resulted in the performance improvement. Figure 11(b) illustrates the DET curves ResNet-II model: the model trained with Softmax, the model trained with Softmax with spectrum augmentation, the model trained with AAM-loss with spectrum augmentation. It could be seen that model with augmentation is better at the EER point, but it is slightly worse at a low miss rate.

Comparison with state-of-the-art models is reported in Table 6. In fact, all state-of-the-art models employ data augmentation, very deep architecture with loss functions that encourage inter-class separability and within-individual compactness. This results confirm observed during experiments behaviour: deeper architecture with more filters provides better performance.

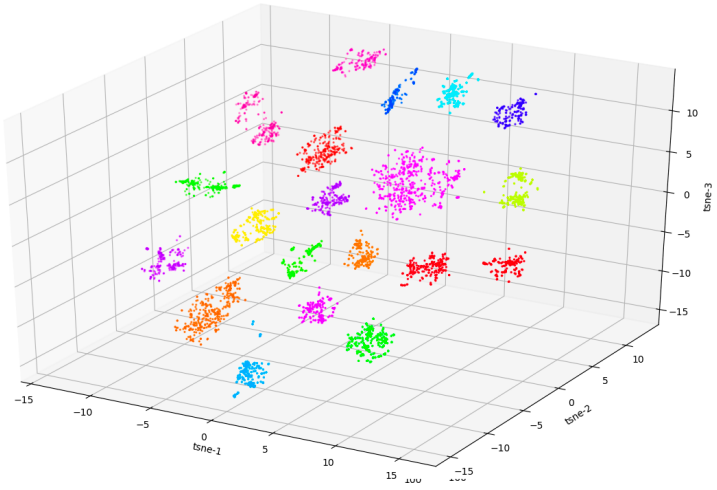
5.3 Visualizing bottleneck features

One of the aim of the project is to investigate the generalization ability of the trained recognition models and to give an inside onto the projection layer responsible for embeddings. Visualization of 2D and 3D projections of activations computed for pre-softmax layer could be the way to understand the nature of identification networks. To visualize 512-dimension feature vectors extracted from a hidden fully-connected layer, t-SNE and PCA are applied for dimensionality reduction. For the t-SNE approach, the number of components, perplexity and training iterations are set to 2, 50 and 1500, respectively. In order to analyse the generalization ability of proposed models, embeddings are obtained on previously unseen train part of VoxCeleb1 data set.

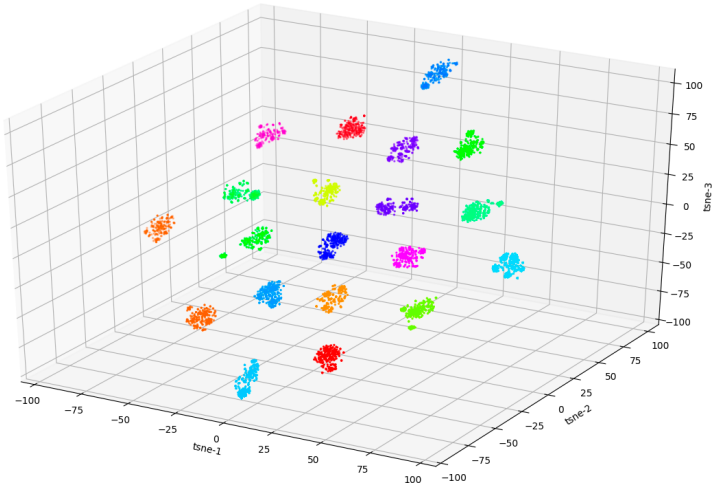
All models provide similar patterns. As an example, 2D and 3D projections of activations sampled from from ResNet34-II pre-softmax layer for 20 previously unseen randomly-selected speakers are shown in Figure 12 and 13. Illustrated results revealed more compact clusters for the system trained on log mel filterbank features. That corresponds to the numerical results presented in Section 5.1-5.2. In fact, this model showed 38% lower error-rate. From Figures 12, it could seen that the t-SNE-based projection allows to separate speakers much better than PCA. Moreover, a combination of these two methods gives even better separation. In fact, t-SNE is applied to dimensionality reduction using PCA. Since cumulative explained variation for 10 principal components is already 0.999, the subspace size is set to 10.

Authors	test set	Model	Loss	EER (%)	C_{det}^{min}
Nagrani *at al.* [29]	VoxCeleb-O	ResNet50 (aug)	contrastive loss	3.95	0.429
Zeilani *at al.*[53]	VoxCeleb-O	ResNet160+cos+S-norm (aug)	AAM loss	1.31	0.154
Xiang *at al* [28]	VoxCeleb-O	TDNN+PLDA (aug)	AAM loss	2.694	-
Xie *at al* [25]	VoxCeleb-O	Thin ResNet34 + GhostVLAD(aug)	Softmax	3.22	-
our model	VoxCeleb-O	ResNet34-II+PLDA (aug)	Softmax	**4.67**	0.478
Nagrani *at al.* [29]	VoxCeleb-E	ResNet50 (aug)	contrastive loss	4.42	0.524
Zeilani *at al.*[53]	VoxCeleb-E	ResNet256+cos+S-norm (aug)	AAM loss	1.35	0.164
Xiang *at al* [28]	VoxCeleb-E	TDNN+PLDA (aug)	AAM loss	2.764	-
Xie *at al* [25]	VoxCeleb-E	Thin ResNet34 + GhostVLAD(aug)	Softmax	3.13	-
our model	VoxCeleb-E	ResNet34-II+PLDA (aug)	Softmax	**4.58**	0.493
Nagrani *at al.* [29]	VoxCeleb-H	ResNet50 (aug)	contrastive loss	7.33	0.673
Zeilani *at al.*[53]	VoxCeleb-H	ResNet256+cos+S-norm (aug)	AAM loss	2.48	0.233
Xiang *at al* [28]	VoxCeleb-H	TDNN+PLDA (aug)	AAM loss	4.732	-
Xie *at al* [25]	VoxCeleb-H	Thin ResNet34 + GhostVLAD(aug)	Softmax	5.06	-
our model	VoxCeleb-H	ResNet34-II+PLDA (aug)	Softmax	**7.91**	0.644

Table 6: Comparison of our models achieved the best performance with state-of-the-art systems. Results are reported on three VoxCeleb1 evaluation lists. All models were trained on VoxCeleb2 dataset only. Since some authors did not published minDCF estimation, this information is missed in the table.

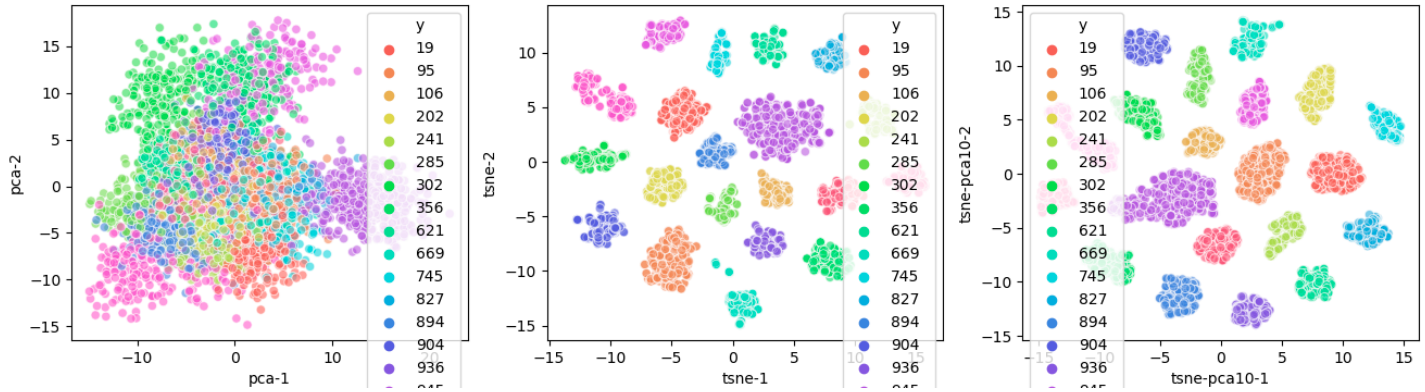


(a) Embedding obtained from NN trained on MFCCs features

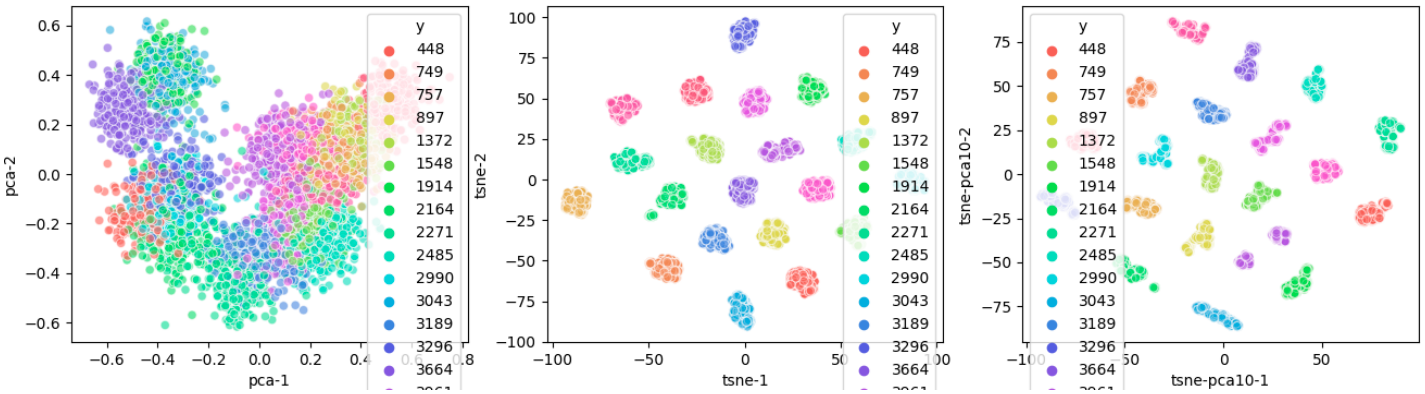


(b) Embedding obtained from NN trained on log mel filterbanks fetures

Figure 12: Visualisations of t-SNE-based projections of embeddings extracted from ResNet34-II fc7 layer. 20 clusters corresponds to 20 different randomly-chosen speakers from VoxCeleb1 train set.



(a) Embedding obtained from NN trained on MFCCs features



(b) Embedding obtained from NN trained on log mel filterbanks features

Figure 13: Visualisations of PCA-based, t-SNE-based, t-SNE+PCA-based projections (1st vs. 2nd dimension) of embeddings extracted from the ResNet34-II fc7 layer. Y denotes a speaker class number.

6 Discussion

An initial objective of the project was to examine NN topology, losses, backends and frame-level input features utilized for text-independent speaker recognition systems. The most obvious finding to emerge from the analysis is a dependence of the performance on a train corpus size. Experiments also confirms that training on longer utterances improves significantly a recognition model. One of the objectives of this thesis was to investigate how performance of recognition systems depends on network capacity. The results of this study indicate that evaluation metrics tend to decrease with network depth and number of filters for ResNet-based models. However, total number of parameters is less important than architecture. In fact, all ResNet-based systems outperformed the plain baseline model even though it contained more parameters. One unanticipated finding was that models fine-tuned with AAM-loss achieved higher error-rate than trained with simple Softmax. This results are contrary to previous studies which have suggested that margin losses encourage inter-class separability. A possible explanation for this might be a slow convergence of AAM-loss. Thus, training with more epochs with a small learning rate could improve results. Prior studies have noted the importance of centering embeddings before scoring. Indeed, mean subtraction constantly improved results for all backends. The results also show that a separately trained PLDA backend model benefits in comparison with cosine similarity scoring. Comparison of the findings with those of other studies confirms that NN model trained on log filterbanks features shows better performance. However, this comparison should be done more accurate by preserving length of input utterances. The most interesting finding was the ability of bottleneck layer to cluster unseen speakers by visualizing activations of the projection layer.

Overall, a recognition model proposed in this study showed good results on all evaluation lists even though there is a gap in the performance between presented and state-of-the-art systems. Worse results may be explained by deeper architectures used in these studies, with more filters in each ResNet block. It is also common for all state-of-the-art systems to apply data augmentation in the time domain by adding noise, background music or reverberation while this study employs spectrum augmentation. Further research should be undertaken to investigate the performance using data augmentation in the time domain, other margin losses and calibration for PLDA scoring.

7 Conclusion

The purpose of the present research was to investigate CNN-based systems for text-independent speaker recognition. Although the current study is based on comparison of only four models, the findings suggest that performance correlate with CNN capacity. Processing time did not allow to train each model with more epochs and examine deeper architectures with larger filter size. A further research could assess whether augmentation in a time domain along with spectrum augmentation improve performance of the recognition system. The second aim of this study was to investigate the ability of the bottleneck layer to separate new speakers. Sampling activation from the embedding layer and projecting on a plain revealed meaningful clusters that confirmed the generalization ability of the model to define speaker manifolds.

8 Acknowledgments

I would like to thank my supervisor, dr. David van Leeuwen, for providing guidance for me over my research, for his patience, motivation, and immense knowledge. And a

special thank you for providing access to the cluster and for assistance with a queuing system. I would like to acknowledge dr. Louis ten Bosch as the second reader of this thesis, and I am gratefully indebted to his for his very valuable comments on this thesis. I am really grateful to have been their student. I would also like to thank dr. Oldřich Plchot for helpful discussions.

References

[1] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end text-dependent speaker verification. *In Acoustics, Speech and Signal Processing (ICASSP) IEEE International Conference on, pages 5115–5119. IEEE*, 2016.

[2] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu. Deep speaker: an end-to-end neural speaker embedding system. *arXiv preprint arXiv:1705.02304*, 2017.

[3] Simon Prince and James Elder. Probabilistic linear discriminant analysis for inferences about identity. *IEEE 11th International Conference on Computer Vision*, pages 1–8, 01 2007. doi: 10.1109/ICCV.2007.4409052.

[4] Luciana Ferrer and Mitchell McLaren. A discriminative condition-aware backend for speaker verification. *arXiv:11911.11622*, 2019.

[5] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur. Deep neural network embeddings for text-independent speaker verification. pages 999–1003, 08 2017. doi: 10.21437/Interspeech.2017-620.

[6] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *arXiv:1604.00289*, 2016.

[7] Hagai Aronowitz, Ron Hoory, Jason Pelecanos, and David Nahamoo. *New Developments in Voice Biometrics for User Authentication.*, pages 17–20, 01 2011.

[8] Yuxin Wang, Qiang Wang, Shaohuai Shi, Xin He, Zhenheng Tang, Kaiyong Zhao, and Xiaowen Chu. Benchmarking the performance and energy efficiency of ai accelerators for ai training. *arXiv: 1909.06842*, 2019.

[9] N. Dehak V. Gupta P. Kenny, P. Ouellet and P. Dumouchel. A study of inter-speaker variability in speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 980–988, 2008.

[10] Najim Dehak, R. Dehak, James Glass, Douglas Reynolds, and Patrick Kenny. Cosine similarity scoring without score normalization techniques. *Odyssey Speaker and Language Recognition Workshop*, 01 2010.

[11] Patrick Kenny. Joint factor analysis of speaker and session variability: Theory and algorithms. *CRIM, Montreal, CRIM-06/08-13*, 01 2006.

[12] Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech and Language Processing*, 2010.

[13] Ahilan Kanagasundaram, Robbie Vogt, David Dean, Sridha Sridharan, and Michael Mason. I-vector based speaker recognition on short utterances. *In Proceedings of Interspeech*, 2011.

[14] Cohen et al. Applied multiple regression/correlation analysis for the behavioural sciences 3rd ed. *Taylor Francis Group*, 2003.

[15] F. Bahmaninezhad and J. H. L. Hansen. i-vector/plda speaker recognition using support vectors with discriminant analysis. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5410–5414, 2017.

[16] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Moreno, and Javier Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. pages 4052–4056, 05 2014. ISBN 978-1-4799-2893-4. doi: 10.1109/ICASSP.2014.6854363.

[17] Yu hsin Chen, Ignacio Lopez Moreno, Tara Sainath, Mirkó Visontai, Raziel Alvarez, and Carolina Parada. Locally-connected and convolutional neural networks for small footprint speaker recognition. In *Interspeech*, 2015.

[18] Desh Raj, David Snyder, Daniel Povey, and Sanjeev Khudanpur. Probing the information encoded in x-vectors. *arXiv: 1909.06351*, 2019.

[19] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification. *arXiv:1710.10467*, 2017.

[20] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: a large-scale speaker identification dataset. *CoRR*, abs/1706.08612, 2017. URL `http://arxiv.org/abs/1706.08612`.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[23] Daniel Garcia-Romero, Greg Sell, and Alan Mccree. MagNetO: X-vector Magnitude Estimation Network plus Offset for Improved Speaker Recognition. In *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*, pages 1–8, 2020. doi: 10.21437/Odyssey.2020-1. URL `http://dx.doi.org/10.21437/Odyssey.2020-1`.

[24] Shuai Wang, Johan Rohdin, Oldrich Plchot, Lukas Burget, Kai Yu, and Jan Cernocky. Investigation of specaugment for deep speaker embedding learning. pages 7139–7143, 05 2020. doi: 10.1109/ICASSP40776.2020.9053481.

[25] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Utterance-level aggregation for speaker recognition in the wild. *arXiv:1902.10107*, 2019.

[26] Yujie Zhong, Relja Arandjelović, and Andrew Zisserman. Ghostvlad for set-based face recognition. *arXiv:1810.09951*, 2018.

[27] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. *arXiv:1511.07247*, 2015.

[28] Xu Xiang, Shuai Wang, Houjun Huang, Yanmin Qian, and Kai Yu. Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition. *arXiv:1906.07317*, 2019.

[29] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv:1806.05622*, 2018.

[30] Chunlei Zhang, Kazuhito Koishida, and John Hansen. Text-independent speaker verification based on triplet convolutional neural network embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26:1–1, 04 2018. doi: 10.1109/TASLP.2018.2831456.

[31] G. Chen, C. Parada, and G. Heigold. Small-footprint keyword spotting using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091, 2014.

[32] M. Magimai-Doss D. Palaz and R. Collobert. Analysis of cnn-based speech recognition system using raw speech as input. *in Proc. of Interspeech*, 2015.

[33] Mirco Ravanelli and Yoshua Bengio. Speech and speaker recognition from raw waveform with sincnet. *arXiv:1812.05920*, 2018.

[34] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. pages 5329–5333, 04 2018. doi: 10.1109/ICASSP.2018.8461375.

[35] Aminadabe Soares, W.D. Parreira, Everton Souza, Sérgio Almeida, Cláudio Diniz, Chiara Nascimento, and Matheus Stigger. Energy-based voice activity detection algorithm using gaussian and cauchy kernels. pages 1–4, 02 2018. doi: 10.1109/LASCAS.2018.8399936.

[36] Abdel rahman Mohamed. Deep neural network acoustic models for asr. *Signal Processing Magazine, 29(6):82–97*, 2014.

[37] Nasser Kehtarnavaz and Namjin Kim. *Digital Signal Processing System-Level Design Using LabVIEW*. Newnes, USA, 2005. ISBN 075067914X.

[38] David Robinson and Malcolm Hawksford. Psychoacoustic models and non-linear human hearing. *The University of Essex. UK*, 01 2000.

[39] Douglas O'Shaughnessy. Speech communication: human and machine. *Addison-Wesley. p. 150. ISBN 978-0-201-16520-3.*, 1987.

[40] Todor Ganchev, Nikos Fakotakis, and Kokkinakis George. Comparative evaluation of various mfcc implementations on the speaker verification task. *Proceedings of the SPECOM*, 1, 01 2005.

[41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

[42] Daniel S. Park, William Chan2, Yu Zhang1, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk1, and Quoc V. Le1. Specaugment: A simple data augmentation method for automatic speech recognition. *INTERSPEECH*, 2019.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618.

[44] Fred Richardson, Douglas Reynolds, and Najim Dehak. A unified deep neural network for speaker and language recognition. *arXiv:1504.00923*, 2015.

[45] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *arXiv:1801.07698*, 2018.

[46] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. *arXiv:1801.09414*, 2018.

[47] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. *arXiv:1704.08063*, 2017.

[48] Itay Salmun, Irit Opher, and Itshak Lapidot. On the use of plda i-vector scoring for clustering short segments. *Odyssey*, 2016.

[49] Garcia-Romero, D., and Espy-Wilson. Analysis of i-vector length normalization in speaker recognition systems. *Proceedings of Interspeech*, (pp. 249–252.), 2016.

[50] J. N Holmes and Wendy Holmes. *Speech synthesis and recognition.* London ; New York : Taylor Francis, 2nd ed edition, 2001. ISBN 0748408568 (hbk.). Previous ed.: Wokingham : Van Nostrand Reinhold, 1988.

[51] Niko Brümmer. Measuring, refining and calibrating speaker and language information extracted from speech. *Ph.D. thesis, University of Stellen- bosch, Stellenbosch, South Africa, Dec.*, 2010.

[52] Brian McFee, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. pages 18–24, 01 2015. doi: 10.25080/Majora-7b98e3ed-003.

[53] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plchot. But system description to voxceleb speaker recognition challenge 2019. *arXiv:1910.12592*, 2019.